

# VisComposer 八月份进展

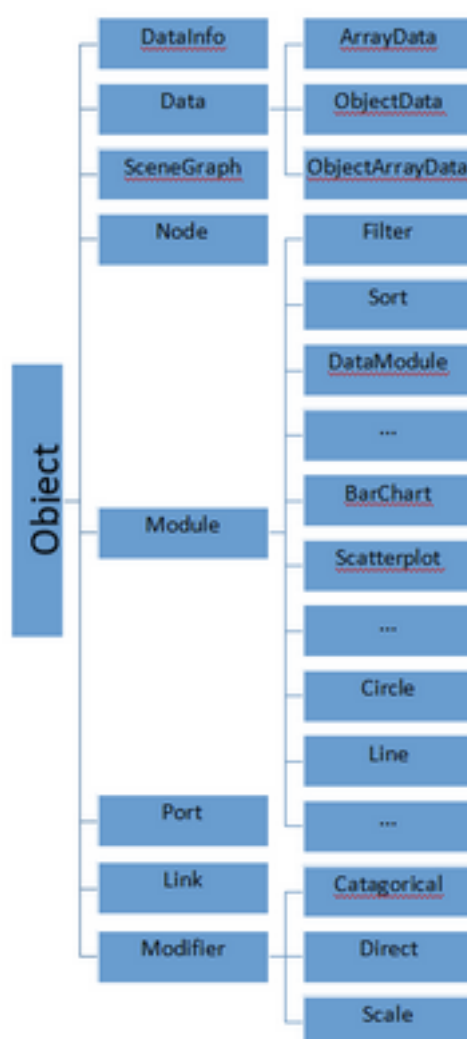
本月工作是与李逸婷一起重构代码。主要分成了两部分：

## 1、优化冗余代码：

之前的版本的代码里，每一种同类型的类都有一个单独的文件，这些类其实有很多共同点，有一样的函数、一样的成员变量。我们做的工作是对这些类抽象出一个基类，不同的基类再抽象出更顶层的基类，把之前的整个工程的架构搭起来。

## 2、系统架构细节修改：

(1) 场景图结构：scenegraph 下面有 node 数组，每个 node 有一个 workflow，每个 workflow 有 input， output， module 和 link，每个 module 下面有 port。下面是代码实现中的类继承结构：



(2) UI 类重构: 之前写的 UI 类跟场景图结构并没有一一对应上, 因此我们都进行了重构, 将一些相同的类抽象出基类并实现继承机制。另外前端静态的 `html dom` 元素也根据场景图结构和重构后的 UI 类结构进行相应的修改, 做到场景图—UI—`html dom` 完全能一一对应上。

(3) 更新机制: `VisComposer` 系统是一个单页应用, 很多的 `ui` 都是动态添加的, 所以系统要运行起来必须要有一个比较完善的基于 `scenegraph` 结构的更新机制。更新的过程是指先自己更新属性和 `ui`, 再更新子类的属性和 `ui` 并一直递归下去。更新机制是深度优先递归的: 每次由 `scenegraph` 先更新, 再按照树结构的顺序更新节点 `node`, 每个 `node` 更新自己的 `workflow`, `workflow` 按照 `module` 连接的拓扑顺序更新所有 `module`, 逐个更新自己的 `input`, `output`, 并在所有 `module` 更新好了之后更新所有的 `link`。每个部分的更新都会生成这部分对应的代码并在更新完了之后直接调用编译运行模块运行, 从而进行绘制图形的更新 (现在用的是 `d3` 绘制库)。

跟鸿辉之前讨论了, 目前的更新机制是可以用的, 但是要提高下运行效率, 比如不用每次操作都进行全部的更新, 可以只调用部分更新; 另外还有很多细节方面的东西需要修改。

(4) 编译运行模块细节修改: 这个模块主要是用来运行 `scenegraph` 生成的代码的, 运行之前需要准备运行时需要的环境变量、绘制函数等。

以上工作的分工是李逸婷完成第一项, 我负责第二项。我这边之前两周进度慢是因为我不太熟悉场景图结构部分的代码, 特别是场景图的更新机制; 而且只有我一个人在写, 工程规模比较大, 有时候改着改着问题越来越多。我估计下周才能出一个版本。